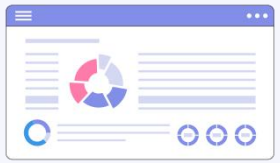
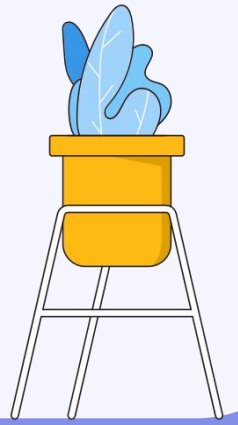


MEDICAL CHARITY SYSTEM REPORT

Prepared By : Muhammed Elbarber



Medical Charity System Report

1. System Overview

The Medical Charity System is a platform designed to manage and streamline the donation of medical equipment, clothes, and medicine for those in need. The system enables efficient tracking, collection, and distribution of donations, ensuring they reach charitable organizations.

Key Actors:

- Donors: Individuals who contribute items.

- Managers: Oversee the entire donation process.

- Agents: Responsible for collecting and delivering donations.

2. System Objectives

- Facilitate Donations: Allow donors to easily contribute items.

- Improve Management: Provide tools for managers to oversee donations, track statuses, and assign agents.

- Efficient Collection: Enable agents to manage donation pickups and track collection statuses.

3. User Roles and Features

Managers

Dashboard: Access a system-wide view of users, donation statuses, and types.

Manage Donations: Accept/reject donations, assign agents, and monitor donation progress.

Oversee Operations: Ensure smooth functioning of the system.

Donors

Dashboard: View the status of donations, categorized by type.

Manage Donations: Submit new donation requests and modify or update existing ones.

Profile Updates: Maintain and update personal information.

Agents

Dashboard: Monitor assigned and collected donation requests.

Task Management: Collect donations from donors, mark them as complete, and review previous tasks.

4. Database Design

User Collection

Fields: firstName, lastName, email, password, gender, address, phone, role, joinedTime
Role: Admin, Donor, or Agent.
Authentication: Users authenticate via email and password, with role-based access.

```
const mongoose = require("mongoose");



const userSchema = new mongoose.Schema({

    firstName: {

        type: String,

        required: true

    },

    lastName: {

        type: String,

        required: true

    },

    email: {

        type: String,

        required: true

    },

    password: {

        type: String,

        required: true

    },

    gender: {

        type: String,

        enum: ["male", "female"]

    },
```

```
    address: String,
    phone: Number,
    joinedTime: {
        type: Date,
        default: Date.now
    },
    role: {
        type: String,
        enum: ["admin", "donor", "agent"],
        required: true
    }
});
```

Equipment Collection

Fields: donor, agent, category, itemDescription, condition, quantity, deliveryTime, address, phone, status, image
Function: Tracks details of donations, including their status and relevant messages.

```
const mongoose = require("mongoose");


const equipmentSchema = new mongoose.Schema({
    donor: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "users",
        required: true
    },
    agent: {
```

```
        type: mongoose.Schema.Types.ObjectId,
        ref: "users",
    },
    category: {
        type: String,
        enum: ["medical equipment", "clothes", "medicine"],
        required: true
    },
    itemDescription: {
        type: String,
        required: true
    },
    condition: {
        type: String,
        required: true
    },
    quantity: {
        type: Number,
        required: true
    },
    deliveryTime: {
        type: Date
    },
    address: {
        type: String,
        required: true
    },
    phone: {
        type: Number,
        required: true
```

```
    },
    collectionTime: {
        type: Date
    },
    donorToAdminMsg: String,
    adminToAgentMsg: String,
    addedAt: {
        type: Date,
        default: Date.now
    },
    status: {
        type: String,
        enum: ["pending", "rejected", "accepted", "assigned", "collected", "available"],
        required: true,
        default: "pending"
    },
    image :{
        type:String,
        required: false
}
});
```

5. System Functions

User Management

Registration and login for managers, donors, and agents.

Role-based access control to ensure proper permissions.

Donation Management

Donors submit donations, view statuses, and track progress.

Managers accept/reject donations and assign tasks to agents.

Agents collect donations and mark them as complete.

Dashboard

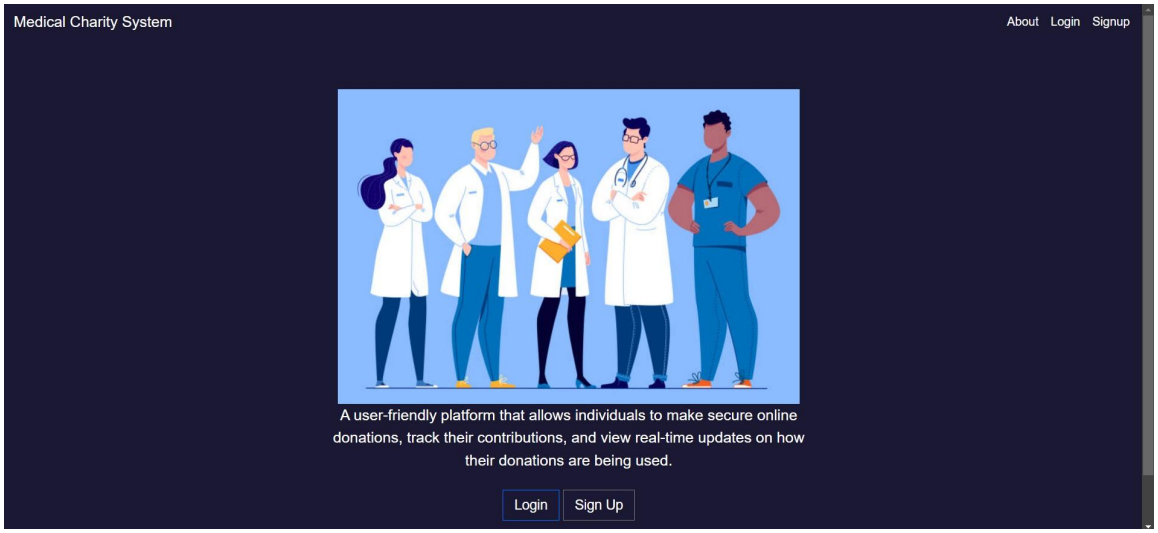
Managers, donors, and agents have personalized dashboards for monitoring activities and donation progress.

Donation Tracking

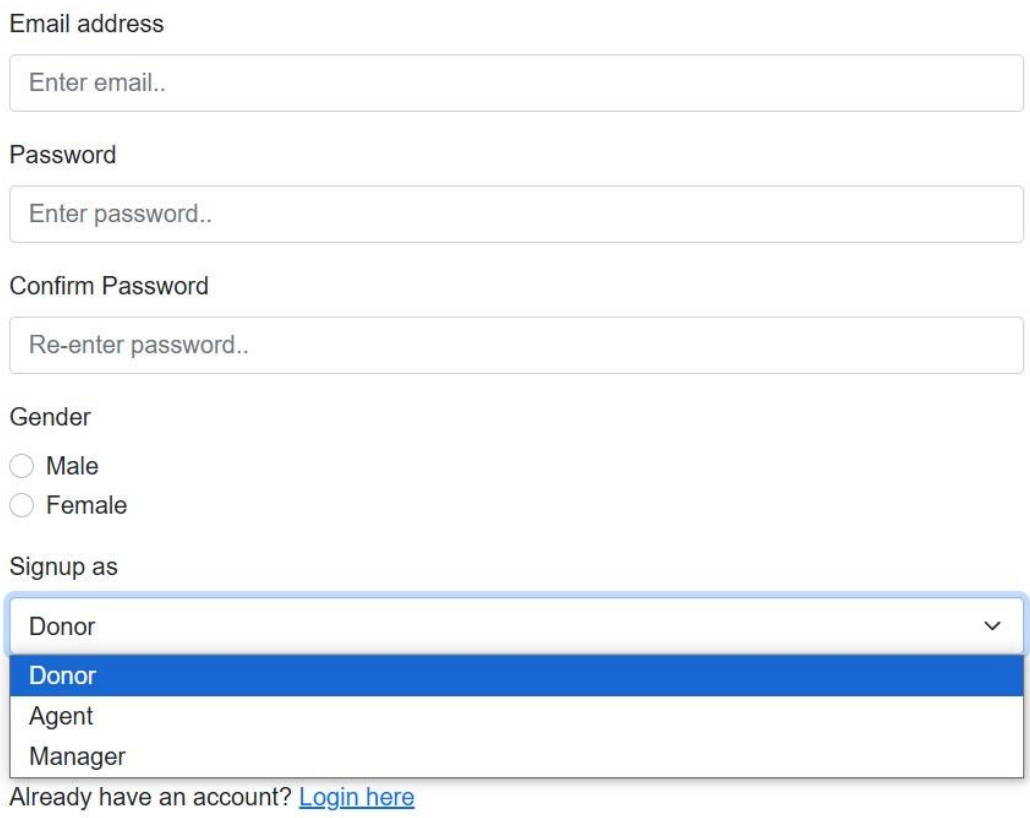
Real-time tracking of donation statuses: pending, accepted, assigned, collected.

6. Main Screens

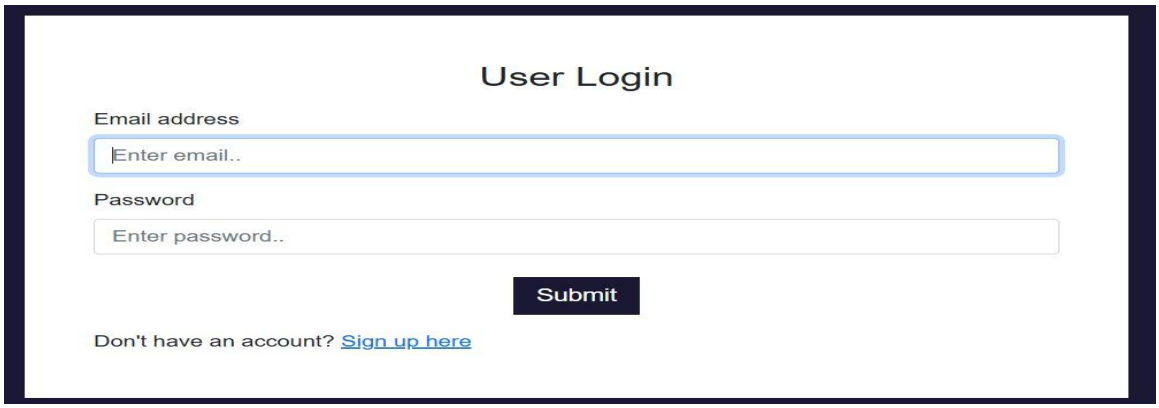
Home Screen



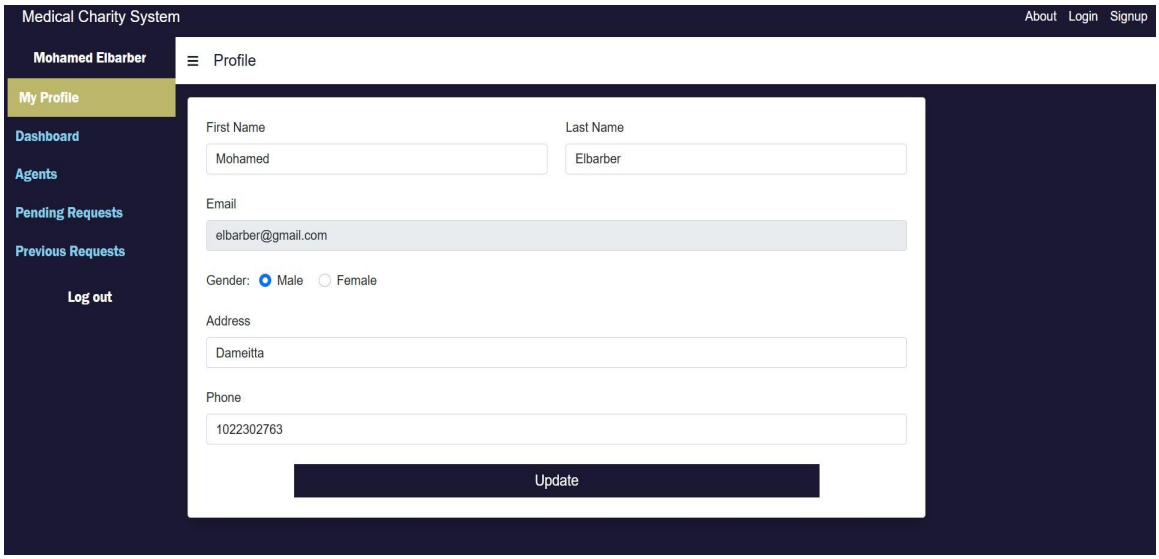
Sign up roles



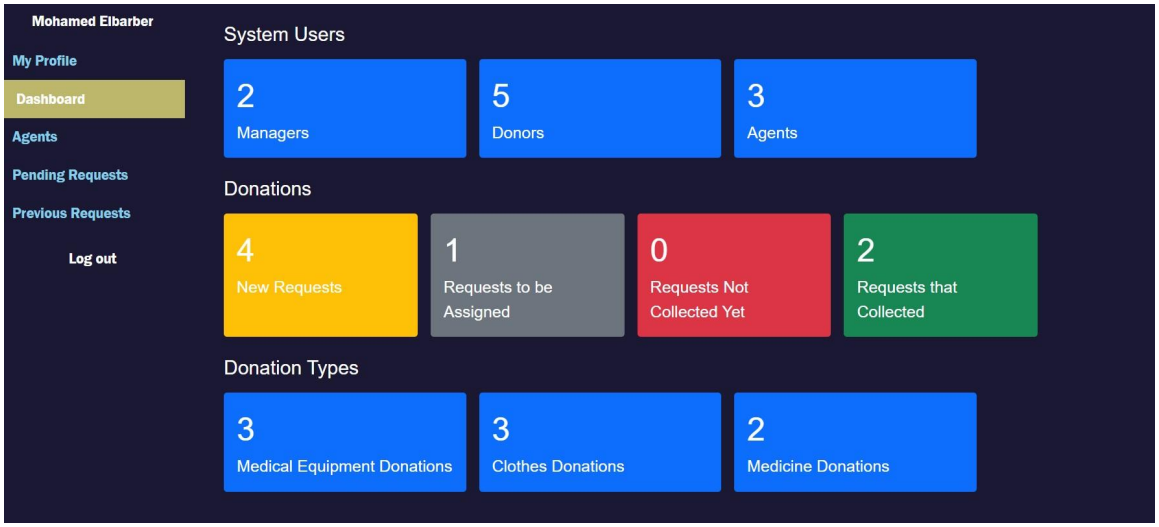
Login Screen



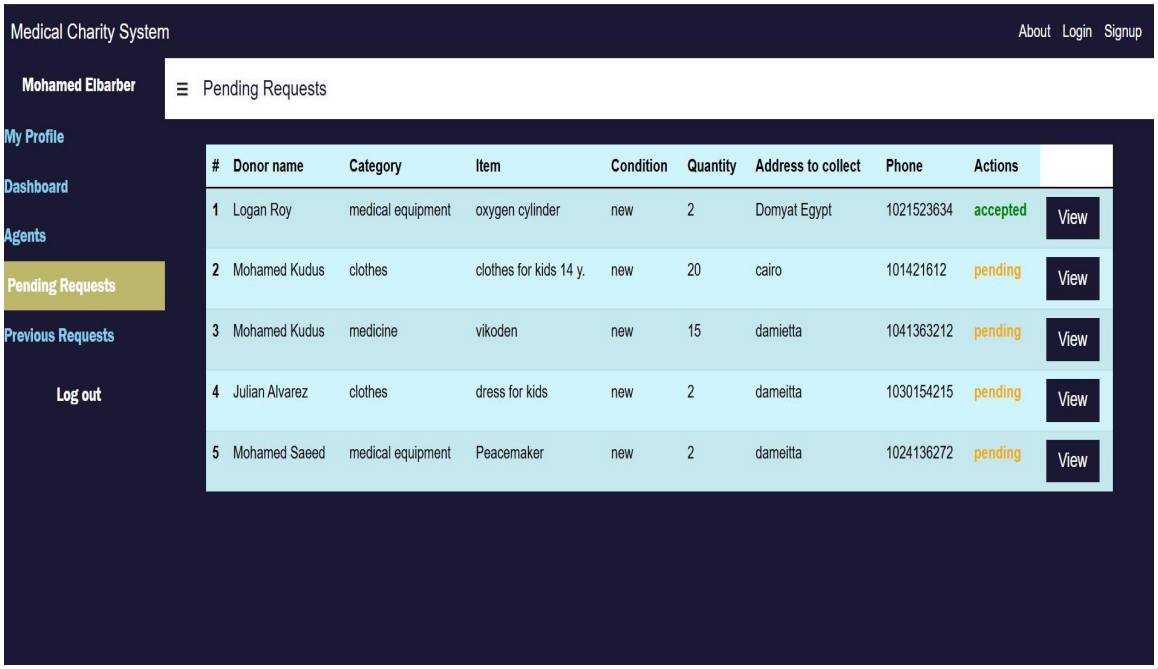
Profile For Users



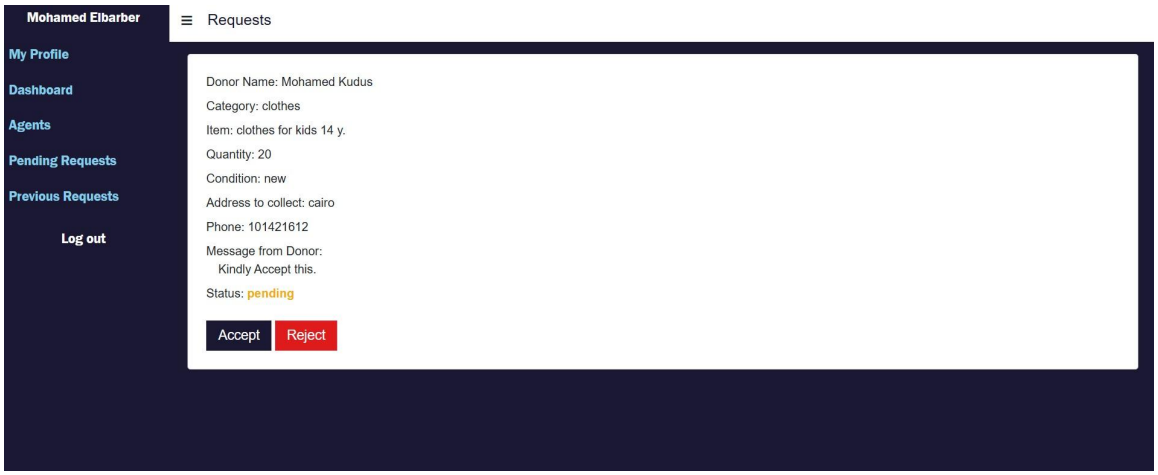
Manager Dashboard



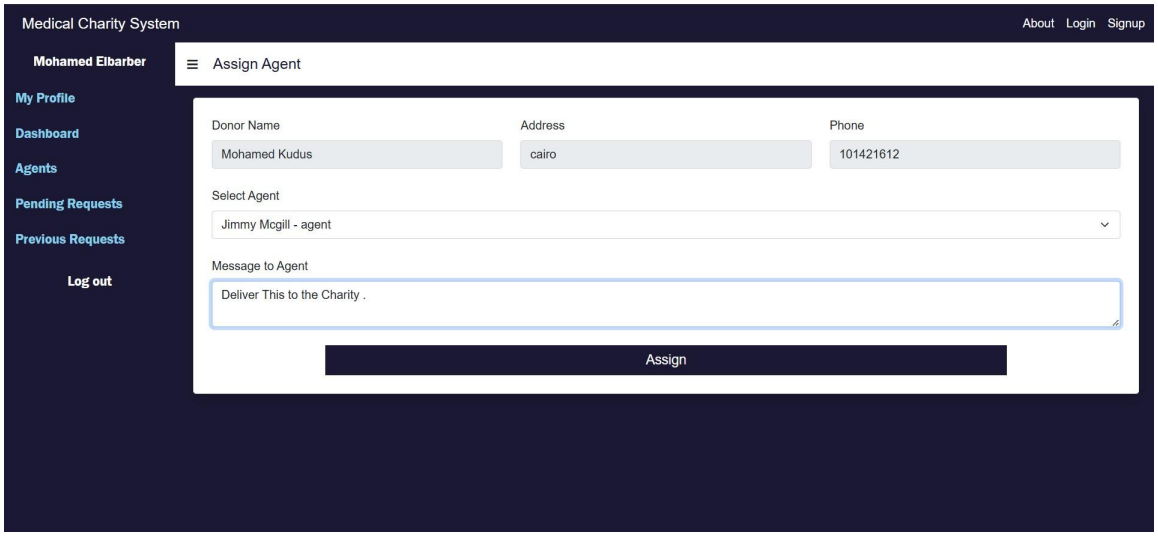
Pending Requests



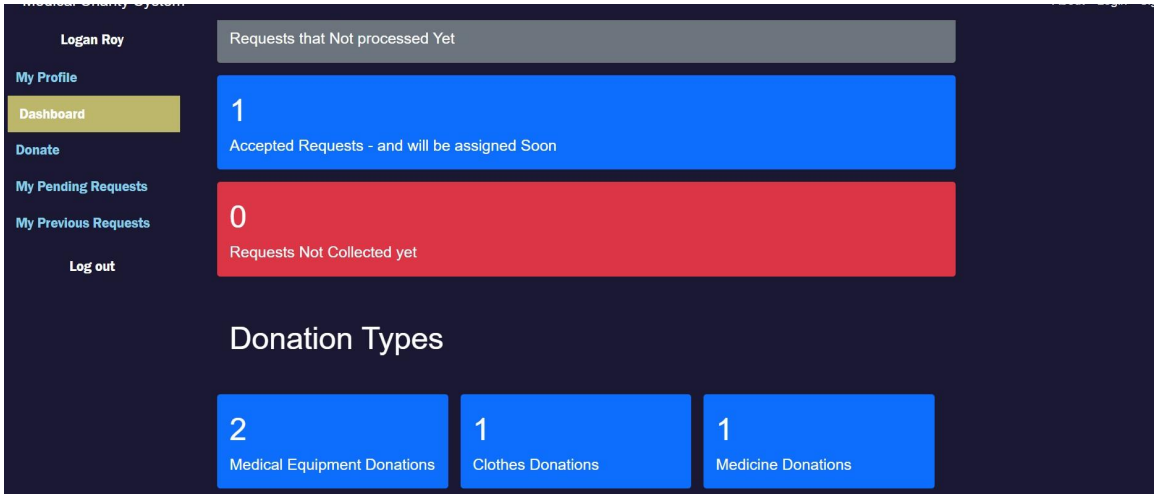
Accept or Reject Donations



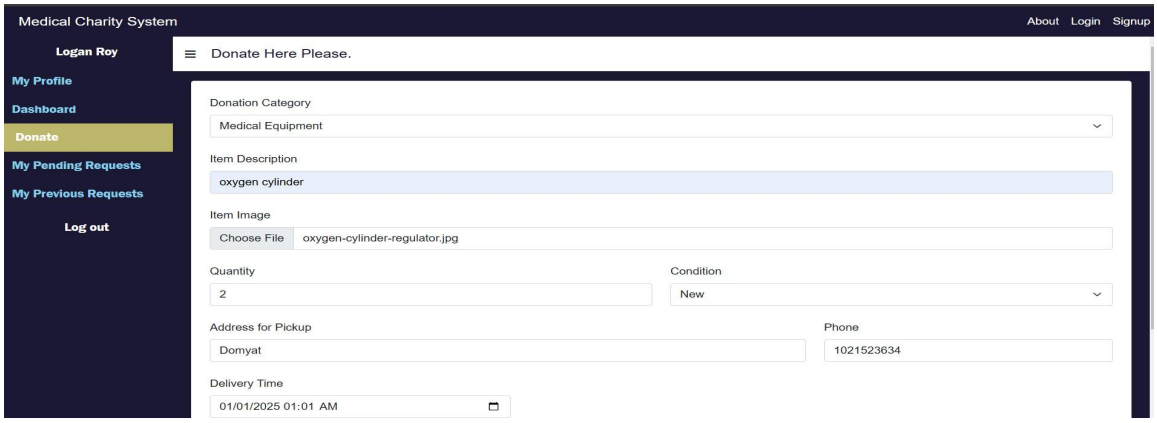
Assign Agent



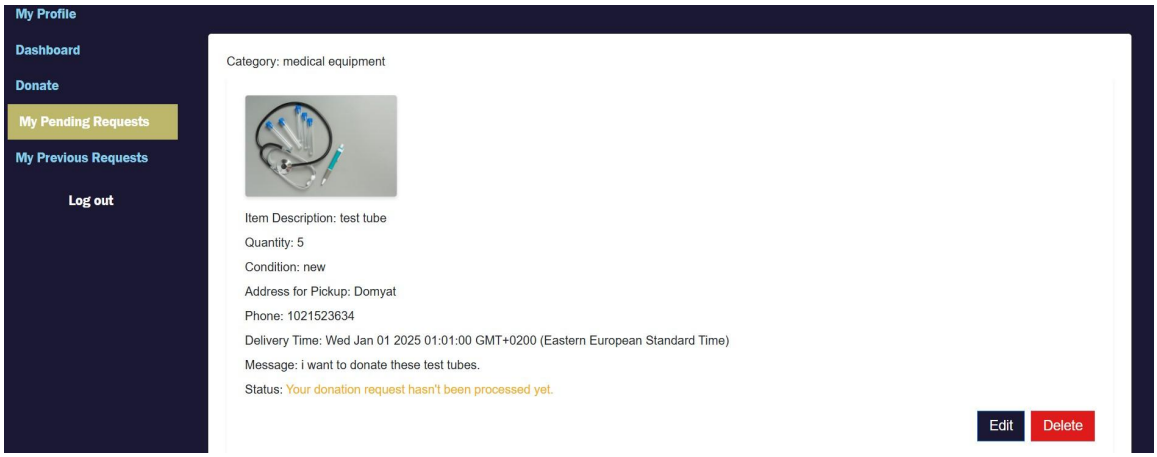
Donor Dashboard



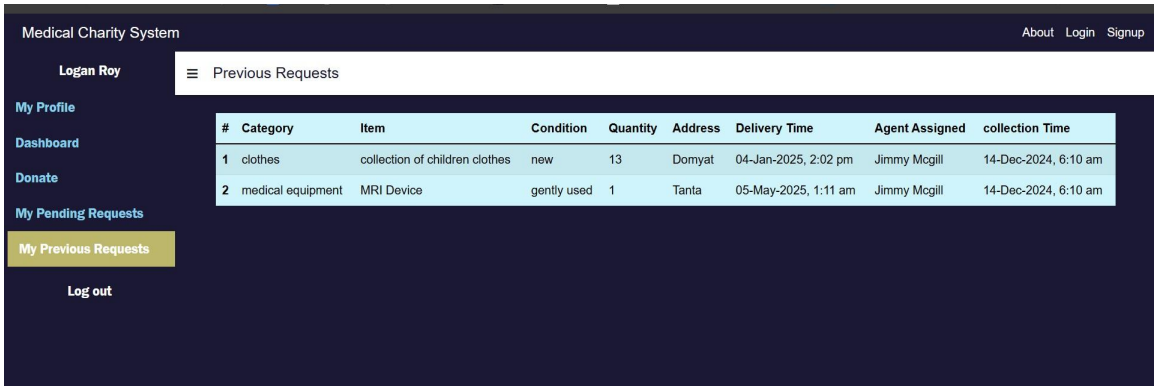
Donate



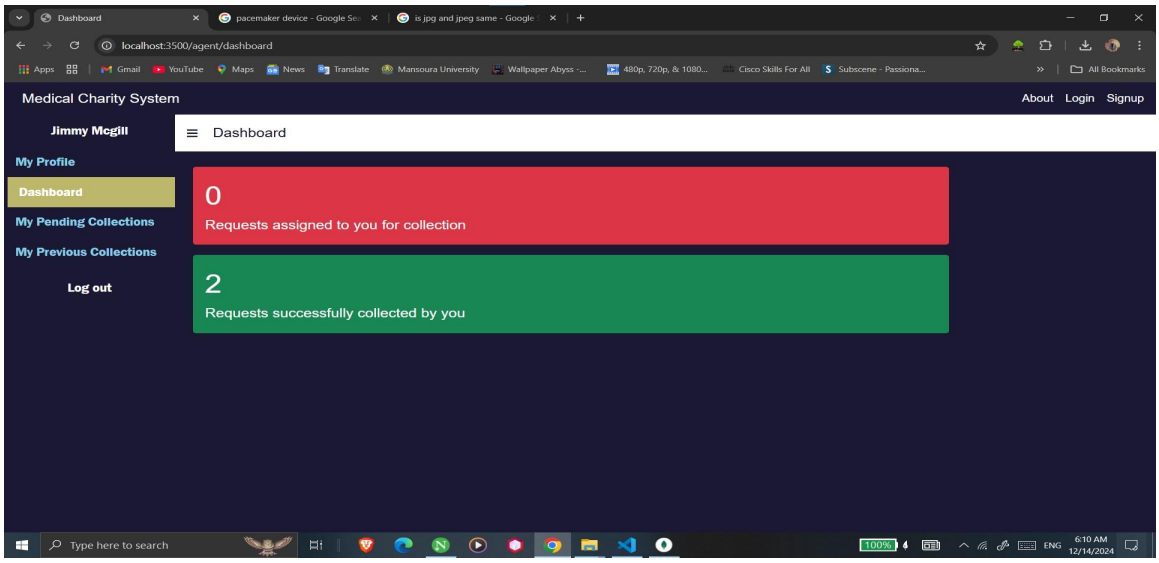
Pending Requests



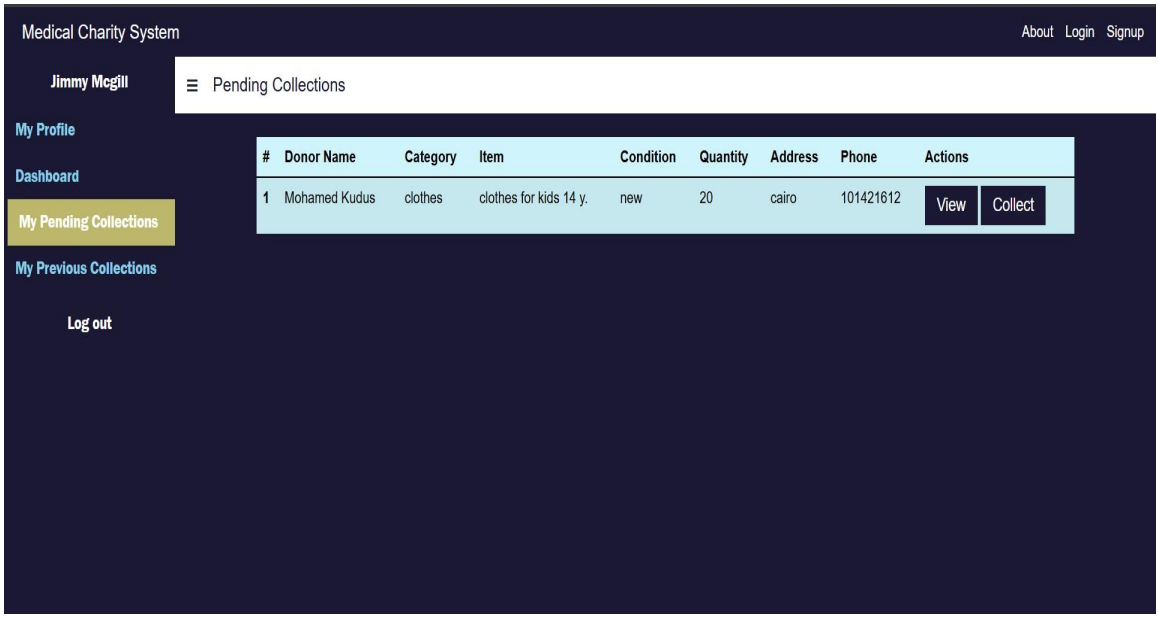
Previous Requests



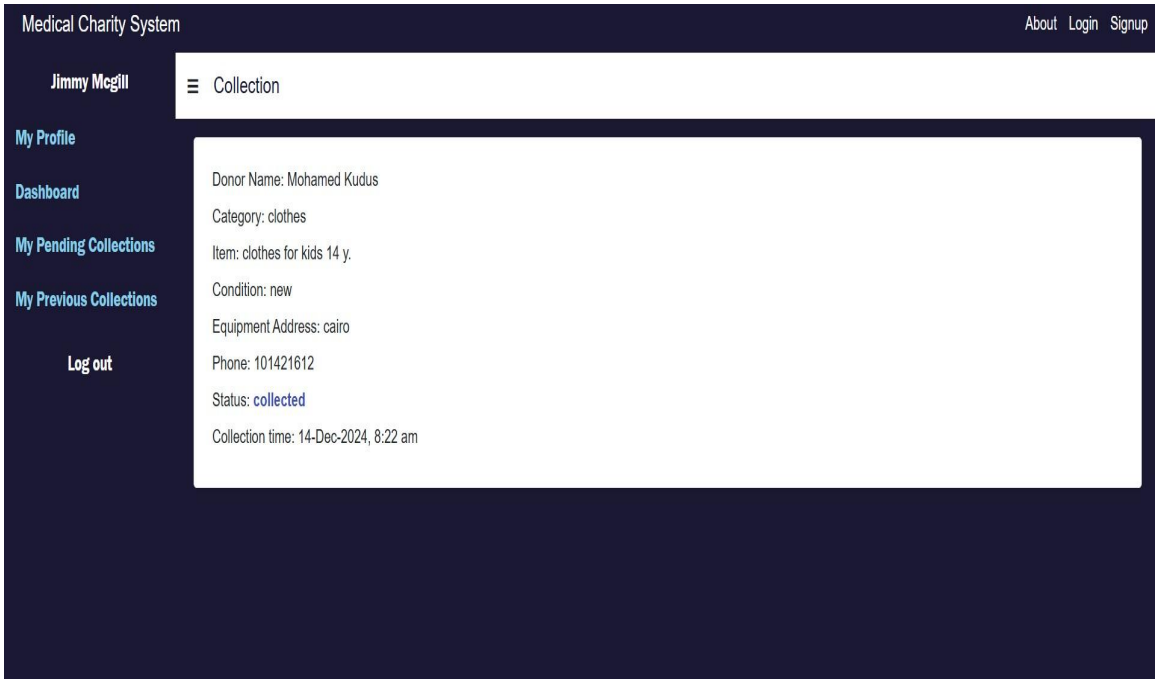
Agent Dashboard



Agent Pending Collections



Collection Status



7. Technology Used

- Node.js: Backend runtime for scalable applications.

- MongoDB: NoSQL database for data storage.

- Express: Web framework for handling requests.

- EJS: Template engine for rendering dynamic HTML views.

- Bootstrap: Frontend framework for responsive design.

- Mongoose: ODM for MongoDB interaction.

- bcryptjs: Password hashing for security.

- passport: User authentication middleware.

- dotenv: Loads environment variables.

- multer: Handles file uploads.

- connect-flash: Provides flash message functionality.

8. Conclusion

The Medical Charity System effectively streamlines the donation process, ensuring that donations are collected and distributed efficiently. The system's structured user roles, robust tracking capabilities, and intuitive dashboards provide transparency and ease of use for all participants.

9. References

1. Node.js Documentation:Node.js. (2024). About Node.js. Retrieved from https://nodejs.org
2. MongoDB Documentation:MongoDB Inc. (2024). MongoDB Developer Manual. Retrieved from https://www.mongodb.com
3. Express Framework Documentation:StrongLoop/IBM. (2024). Express.js Guide. Retrieved from https://expressjs.com
4. EJS Documentation:EJS. (2024). Embedded JavaScript Templates. Retrieved from https://ejs.co
5. Bootstrap Documentation:Bootstrap Team. (2024). Bootstrap 5.3. Retrieved from https://getbootstrap.com
6. Mongoose ODM:Mongoose. (2024). MongoDB ODM for Node.js. Retrieved from https://mongoosejs.com
7. Bcryptjs:bcryptjs. (2024). Password Hashing for Node.js. Retrieved from https://www.npmjs.com/package/bcryptjs
8. Multer for File Upload:Multer. (2024). Handling File Uploads in Node.js. Retrieved from https://www.npmjs.com/package/multer
9. Passport.js:Passport. (2024). Authentication Middleware for Node.js. Retrieved from http://www.passportjs.org